Physical computing from Scratch using scratchClient – **Intermediate**

Control servos, LEDs and more from Scratch using RPi, Arduino, scratchClient

Hans de Jong & Gerhard Hepp

Pi And More 10 1/2

Stuttgart – 24 February 2018

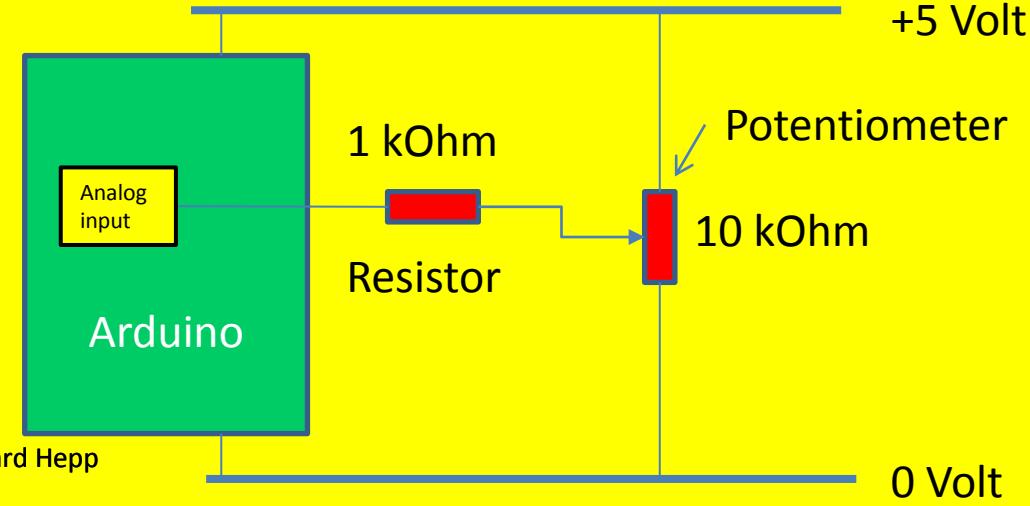
Part 1: Adding analog input

Changing the wiring on the board

- When changing the wiring on the board
  - First pull the USB cable out
  - Switch off the 9V supply

The electronics of a potentiometer

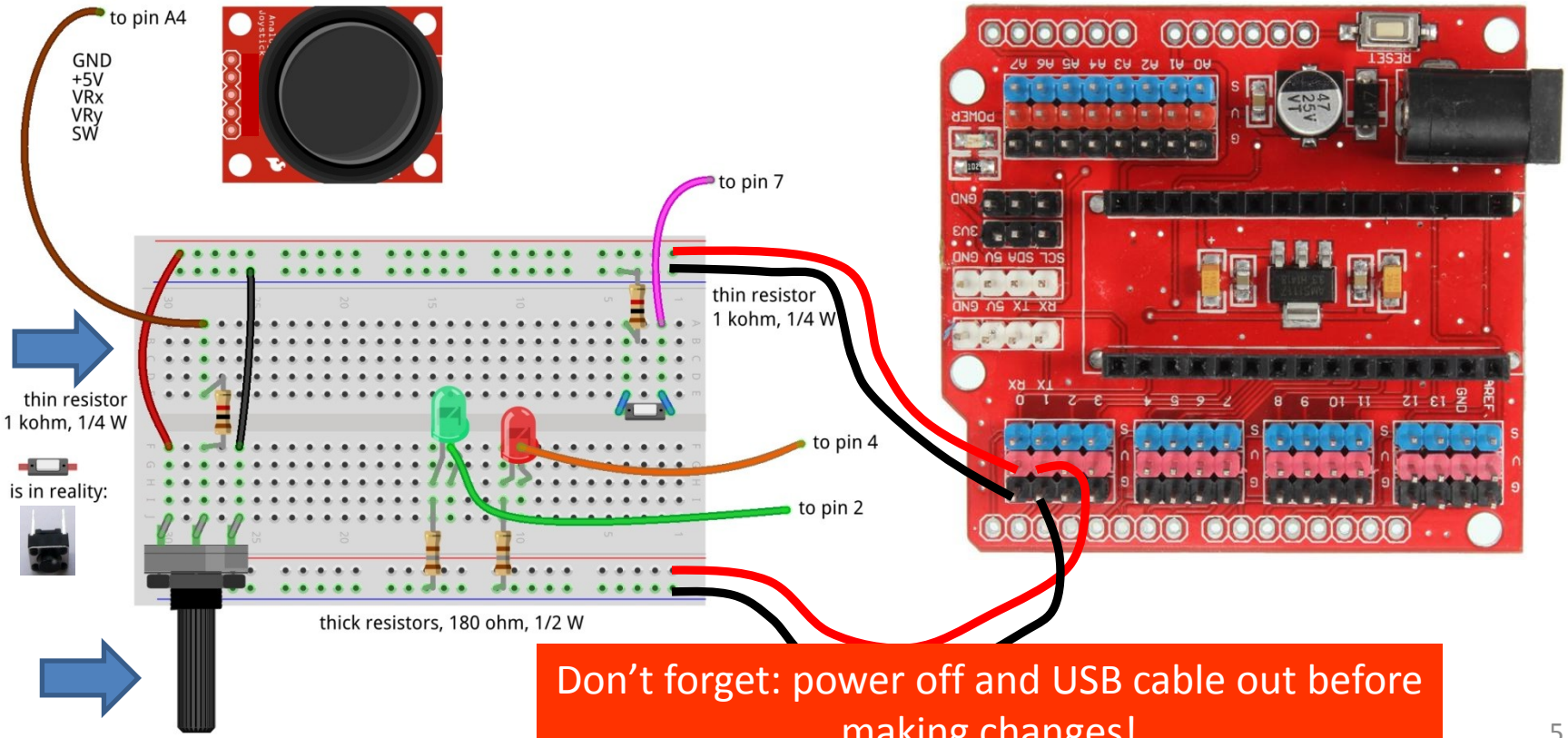
- The potmeter is a resistor that will be connected between 5 V and 0 V.
- Over the length of the resistor the voltage drops linearly.
  - E.g. in the middle it would be 2.5 Volt
  - So this is an analog signal. The value can be changed between 5 V and 0 V.
- There is a 1 kOhm resistor in series for the reasons discussed earlier



Consider what would happen if the 1 kOhm resistor were not there and the port would be used as output and potmeter positioned close to the 0 Volt side.

Then the Arduino would output 5 Volt, and the potmeter is directly leading it to the 0 Volt line → short circuit! Likewise if the potentiometer is close to the 5 Volt side and the Arduino would output 0 Volt.

Adding analog input



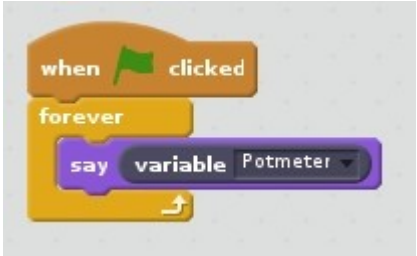
Don’t forget: power off and USB cable out before making changes!

Update the config file

- Update the config file again (right click *PiAndMoreConfig.scl* on the desktop, scratchConfig Edit)
  - Define *Potmeter* on pin A4 (direction: in, function: analog)
- Now that you are updating anyway, already define for the next steps:
  - Define *Servo1* on pin 12 (direction: out, function: servo)
  - Define *BigBlueLED* on pin 5 (direction: out, function: PWM)
  - Define *Buzzer* on pin 11 (direction: out, function: PWM)
- Don’t forget to save. Keep the config editor open.

Test whether it works

- Reconnect and repower the board
- Stop and restart scratchClient by double clicking *PiAndMoreConfig.scl* on the desktop
- Make this piece of program and run it



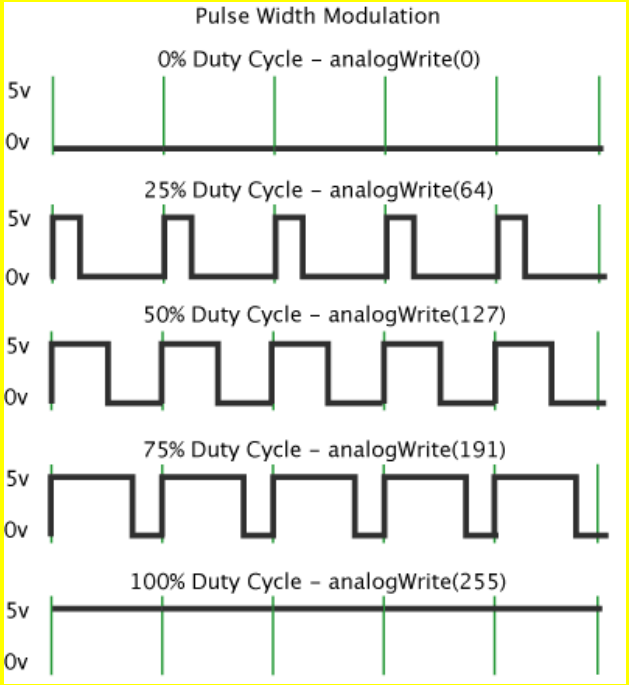
- Turn the knob and see the values of *Potmeter* change
  - Between about 0 and about 1023
- We will write some Scratch program to use it in the next step

Part 2: Pulse Width Modulation

Where we are …

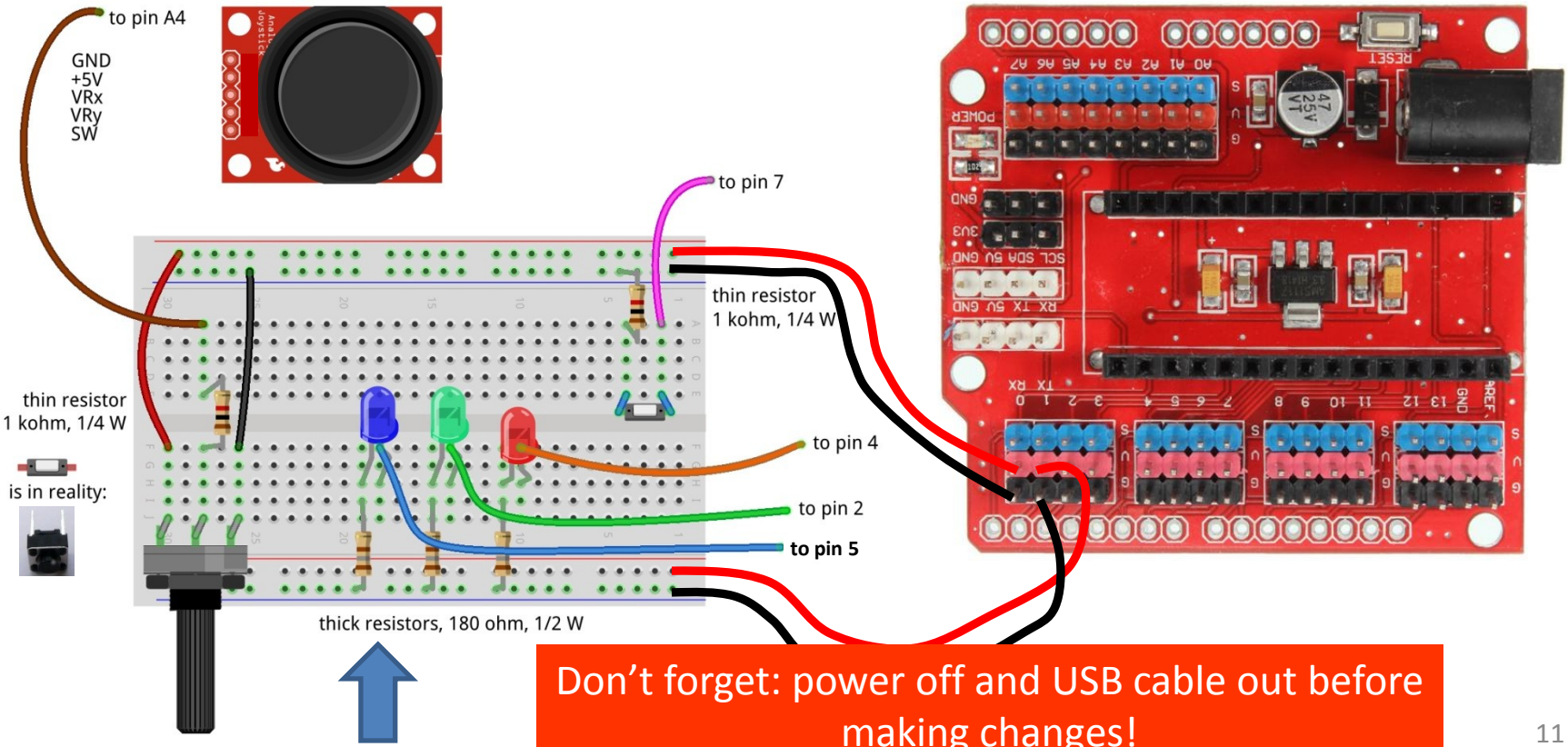
- We have seen:
  - Digital Input (the button) (in the beginners module)
  - Digital Output (the LEDs) (in the beginners module)
  - Analog Input (the potentiometer)
- So what would be most logical next?
  - Analog Output? → but it does not exist!
  - However there is a good alternative: Pulse Width Modulation

Pulse Width Modulation (PWM)



- By modulating (changing) the pulse width, the amount of energy fed to e.g. the LED is changed, and hence the intensity with which you see it lighting.
- Note that in practice the LED is blinking some 800 blinks / second.
- However, no human eye can see more than 100 blinks / second.

Dimming the BigBlueLED with PWM



Don’t forget: power off and USB cable out before making changes!

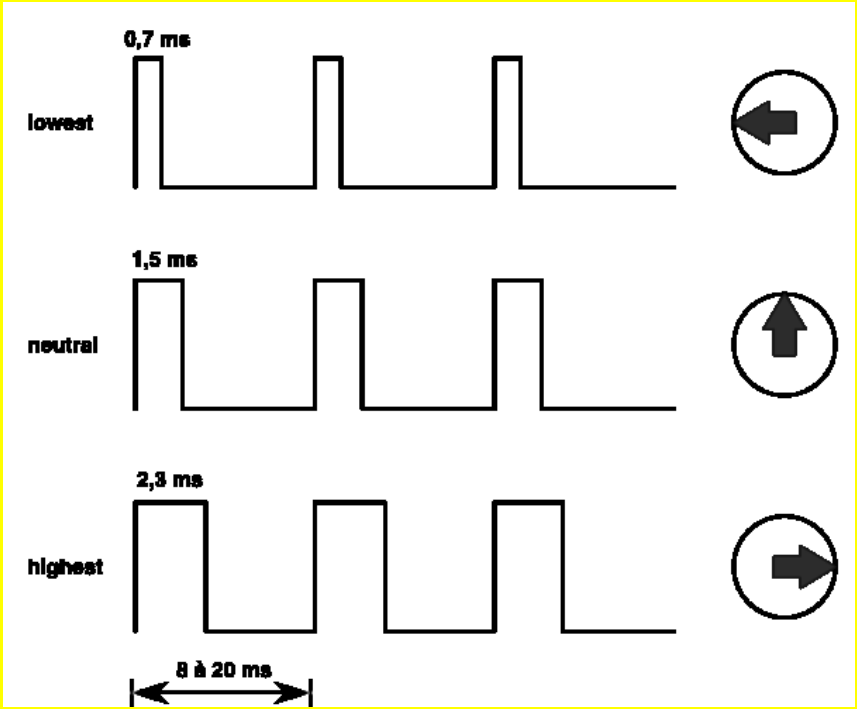
Test

- Switch the power on again and reconnect
- You will see that scratchClient finds the board again
- You already updated the config file in the previous step, so no need to restart scratchClient
- Send *BigBlueLED* values between 0 and 255
- Does the LED brightness change?

Connect Potmeter and Big Blue LED (via Scratch)

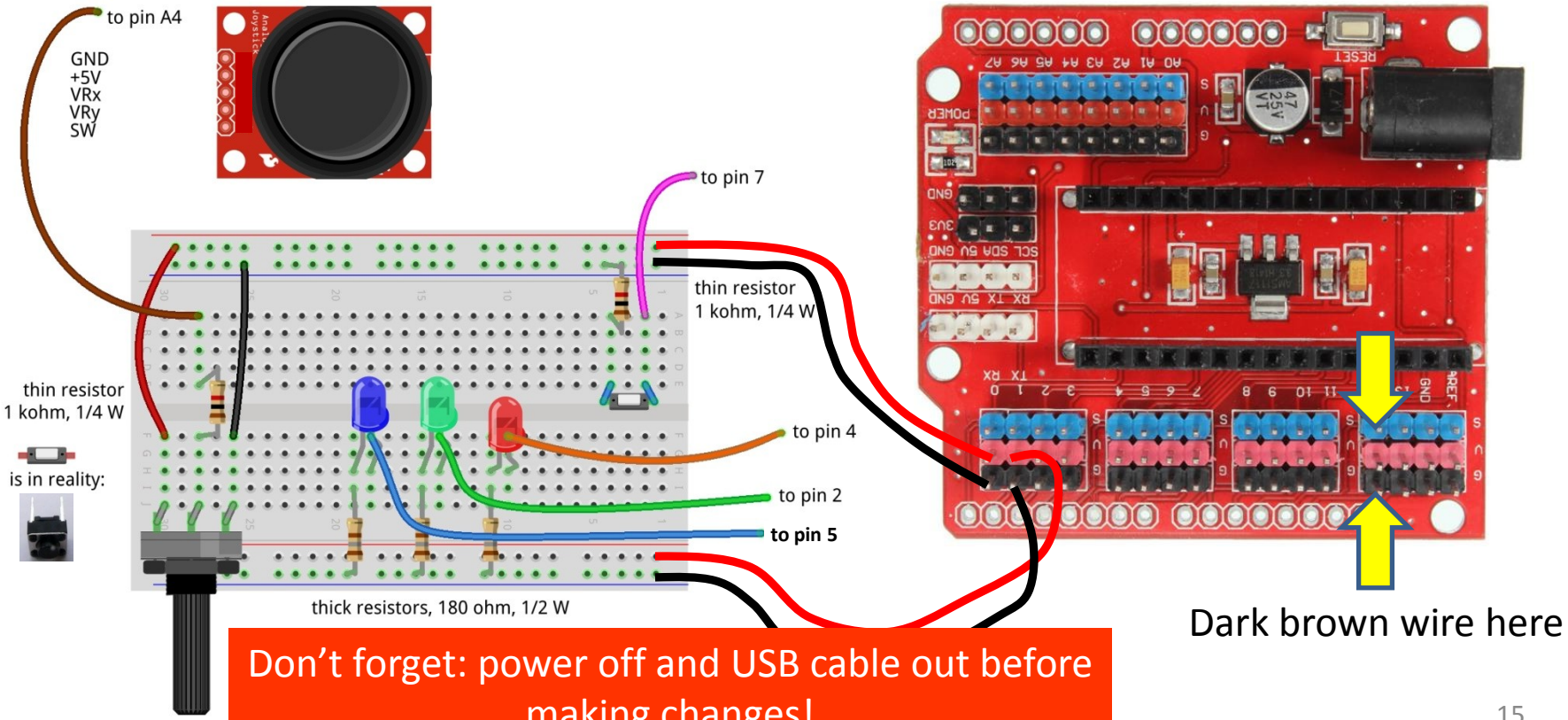
- Make some code that takes the potentiometer reading (between 0 and 1023) and transforms it into the range 0 to 255 (so divide by 4) and set the value of *BigBlueLED*.
- Try out whether turning the potmeter changes the light intensity.
- Notice that when turning to the right, the intensity goes down.
  - You would expect it to go up …
  - How can you very simply change this by interchanging two wires?

Controlling a servo with PWM



- The position of the servo is changed by sending pulses of different width.
- The servo looks at the pulsewidth and turns as desired.
- The servo gets power separately.
- With a servo, the pulse width modulation is not controlling the amount of energy fed to the servo
- With a servo, pulse width modulation is rather a communication protocol.

Adding the servo to pin 12



Don’t forget: power off and USB cable out before making changes!

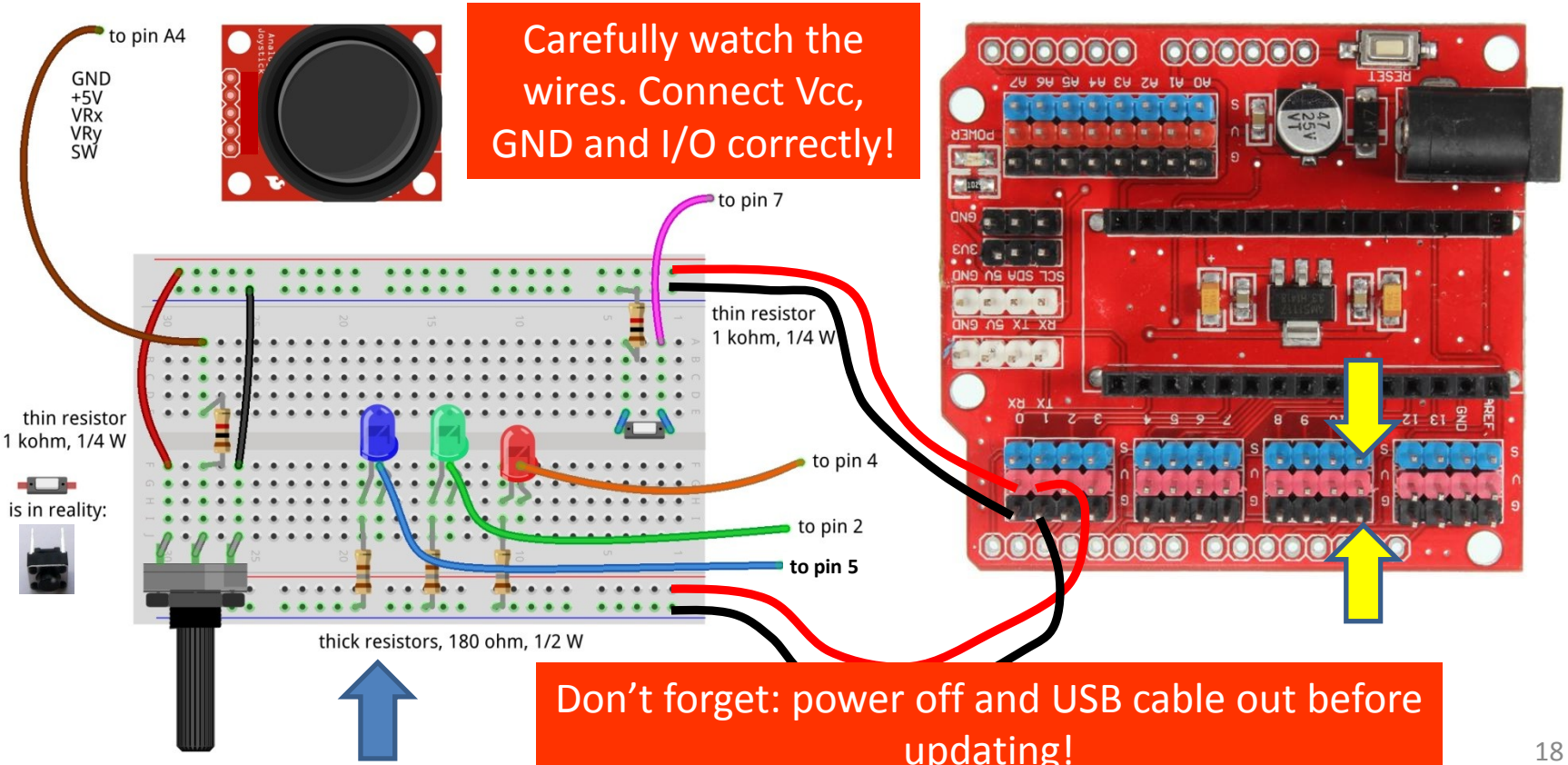
Testing the servo

- Reconnect and repower the board
- Update the loop where you set the *Big Blue LED* value dependent on the potmeter reading to now also set *Servo 1*.
  - But watch out: values have to be between 0 and 180.
  - To avoid overflow: first divide by 1024, then multiply by 180.
- You may observe jitter.
  - Because the values of the potmeter will drift a bit and there will be power fluctuations, the reading of the potmeter will not be constant.
  - We will in the advanced workshop show how to deal with jitter

Controlling a buzzer with PWM

- The buzzer will sound if it gets a signal in an audible frequency range.
- A PWM signal on Arduino gives ca. 800 Hz.
  - Different for different pins
- You will see that there is not much influence by changing the duty cycle.

Connect the buzzer to pin 11



Testing the buzzer

- No need to restart scratchClient, because the config file was not changed.
- Add the setting of the buzzer to the loop where you already set the servo and the Big Blue LED.
  - Like with the LED, the values must be between 0 and 255.
- Only test a short time (to save the ears of your neighbors ☺).

PWM limitations of Arduino (Nano and Uno)

- General PWM is available on pins 3, 5, 6, 9, 10, 11
- Servo can be configured on those pins, but also on 2, 4, 7, 8, 12
- If a servo is configured on any pin, pins 9 and 10 cannot be configured as PWM anymore.
  - The config tool will warn you if you do it wrongly.

Part 11: Summary & take aways

Takeaways of the beginners and intermediate workshops

- With scratchClient you define:
  - Function of each pin
  - Symbolic name for each configured pin
- scratchClient config is the tool to setup the configuration
- Restart scratchClient after you changed the configuration
- Put a resistor in series with LEDs
- Put a resistor in series with switches
- Put a resistor in series with the middle contact of a potentiometer.
- Configure a pull up resistor if the input signal goes between 0 Volt and being open rather than between 0 Volt and 3 to 5 Volt.
- Output signals are controlled from Scratch via the extension blocks
- Input signals are obtained via extension blocks
- You can monitor the value of all pins from the browser
- **scratchClient can do much more…**

- Functions that a pin on Arduino can have:
  - Digital In
  - Digital Out
  - Analog In
  - *No Analog Out*
  - Pulse Width Modulation as alternative for Analog Out
    - For modulating the brightness of a LED
    - For controlling a servo
    - For controlling a buzzer
  - There a few more, see the advanced workshop
  - You can configure pull up resistors on Digital In

End of the **intermediate** workshop